פרויקט גמר בקורס "מבנה המחשב" / תשפ"ד סמסטר ב'

מגישים:

איגור בזוייבסקי ת.ז. 342388717

ירון זילברשטיין ת.ז. 318321759

ענבל אנגל ת.ז. 206860389

מרצה הקורס:

ד"ר גדליה אוקסמן

הוגש בתאריך: 10.8.2024

חלק א'- תיאור הפרויקט

מטרת הפרויקט היא לדמות את אופן הפעולה של מעבד SIMP בעזרת תכנות בשפת C.
הפרויקט מחולק לשני חלקים עיקריים. הראשון הוא האסמבלר, שתפקידו לקרוא קבצי קלט בשפת אסמבלי, לתרגם אותם לשפת מכונה ולייצר תמונת זיכרון התחלתית עבור המעבד. השני הוא הסימולטור, אשר מקבל את מוצא האסמבלר (תמונת זיכרון התחלתית המכילה הוראות לביצוע בשפת מכונה) יחד עם קבצים נוספים, ומבצע את ההוראות לפי הסדר הנדרש. בנוסף, כתבנו קבצי קלט בשפת אסמבלי הבודקים את פעולת המעבד.

הפרויקט מומש באמצעות שני פרויקטים נפרדים בסביבת Visual Studio Community, אחד לכל תוכנה. עבור כל חלק הוגדרו שני קבצים; הראשון, קובץ header המכיל את ההגדרות, קבועים, מבני הנתונים והגדרות של הפונקציות השונות שנעשה בהן שימוש. השני, קובץ source בו מומשה פונקציית main וכל שאר הפונקציות. הרחבה על מטרתן ואופן פעולתן של הפונקציות מופיעה בדוקומנטציה הרלוונטית לכל תוכנה.

המעבד מכיל מערך זיכרון של 16 רגיסטרים פנימיים ברוחב 32 ביטים, כאשר ייעוד כל רגיסטר נקבע בהגדרת הפרויקט. בנוסף למערך הרגיסטרים הפנימיים, המעבד תומך במערך בעל 23 רגיסטרי חומרה שבעזרתם ניתן להתממשק עם התקני קלט/פלט: המעבד תומך בקלט\פלט לפי ההוראות שהוא מקבל ומכיל מספר רכיבים כדי לתמוך בפונקציונליות הנדרשת. המעבד מחזיק בנוסף בזיכרון ראשי ברוחב 20 ביט ובעומק של 4096 שורות. אל המעבד מחובר דיסק קשיח המורכב מ 128 סקטורים, ובכל סקטור יש 128 שורות ברוחב 20 ביט גם כן. בנוסף לדיסק הקשיח יש עוד מספר התקני קלט/פלט - נורות leds, צג 7segment ומסך מונוכרומטי.

חלק ב'- Assembler

תיאור התוכנה:

תפקידו של האסמבלר במעבד SIMP הוא לייצר תמונת זיכרון התחלתית עבור הסימולטור. האסמבלר מתרגם קובץ קלט המכיל הוראות בשפת האסמבלי המוגדרת בפרויקט (לצד תוויות והערות) לקובץ memin.txt המכיל את ההוראות בשפת מכונה ומהוה את תמונת הזיכרון ההתחלתית של המעבד. קובץ זה מכיל גם את זיכרון ההוראות וגם ערכי קבועים שנדרש לשמור בזיכרון, במידת הצורך (נתון כי המעבד תומך בפסודו-הוראה word. אשר מאפשרת כתיבה ישירה לזיכרון).

קלט/פלט:

קלט - קובץ בשפת אסמבלי, פלט - קובץ טקסט המדמה תמונת זיכרון התחלתית.

מבני נתונים שהוגדרו:

Struct	פונקציונליות
label	מבנה לצורך ניהול התוויות. מכיל שלושה שדות: שם התווית, מספר השורה ואינדיקטור המציין אם התווית אותחלה.
instruction	מבנה לצורך ניהול הוראה. מכיל חמישה שדות: סוג האופקוד, רגיסטר יעד, רגיסטר מקור, רגיסטר נוסף ושדה של הקבוע (0 עבור הוראות R-type).
word_inst	מבנה לצורך ניהול הכתיבה לזיכרון בהוראות word. כתובת לכתיבת המידע, הקבוע שנדרש לכתוב, ואינדיקטור המציין אם המבנה אותחל.

פונקציות עזר:

פונקציה	תיאור
Open_file_to_read	פונקציה בסיסית המקבלת שם של קובץ, פותחת אותו ומחזירה פוינטר לקובץ על מנת שנוכל לקרוא ממנו. אנחנו משתמשים בה על מנת לקרוא את קובץ הקלט של התוכנית.

Open_file_to_write	פונקציה בסיסית המקבלת שם של קובץ, פותחת אותו ומחזירה פוינטר לקובץ על מנת שנוכל לכתוב אליו. אנחנו משתמשים בה על מנת לכתוב לקובץ הפלט הנדרש.
Find_label_words_lines	הפונקציה מקבלת פוינטר לקובץ ומערך ריק המוגדר מראש של מילים ושל תוויות. הפונקציה מבצעת מעבר על קובץ הקלט, מסווגת את התוויות והמילים ושומרת אותם במקומות המתאימים במערך.
Contains_label	פונקציית עזר המקבלת מחרוזת ומחזירה אמת אם היא תווית.
Set_lable	פונקציית עזר המוסיפה תווית למערך התוויות.
Write2memin	הפונקציה המרכזית של האסמבלר. מבצעת את פעולת הכתיבה לקובץ הפלט. הפונקציה מקבלת פוינטרים לקבצי הקלט והפלט ואת מערכי המילים והתוויות שנמצאו במעבר הראשון על קובץ הקלט. הפונקציה קוראת כל שורה בקובץ הקלט ובעזרת פונקציות עזר כותבת את השורות המתאימות לקובץ הפלט.
Write_i2memin	פונקציית עזר שנועדה לכתוב שתי שורות לקובץ הפלט במידה ואנחנו מטפלים בהוראה המכילה קבוע, הוראה מסוג I.
Write_r2memin	פונקציית עזר שנועדה לכתוב שורה אחת לקובץ הפלט במידה ואנחנו מטפלים בהוראה שלא מכילה קבוע, הוראה מסוג R.
Find_if_pc_word	פונקציה המקבלת את מערך המילים וPC, ומחזירה אמת אם מדובר בשורה שאמורה להכיל את המידע שנמצא בפקודת הword המתאימה.
Decode_opcode	פונקציית עזר המקבלת אופקוד כמחרוזת, ומחזירה מספר שלם המתאים לסוג האופקוד לפי הגדרת הפרויקט.
Decode_register	פונקציית עזר המקבלת שם רגיסטר כמחרוזת, ומחזירה מספר שלם המתאים למספר הרגיסטר במעבד לפי הגדרת הפרויקט.
Decode_imm	פונקציית עזר המקבלת קבוע כמחרוזת ואת מערך התוויות, הפונקציה בודקת אם מדובר בתווית או בקבוע רגיל, ומחזירה את המידע הרלוונטי כמספר שלם.

convertToDecimal	פונקציית עזר המקבלת מספר בהקסה המיוצג כמחרוזת, ומחזירה את ערכו כמספר שלם דצימלי.
Write_rest_word	פונקציית עזר המקבלת פוינטר לקובץ הפלט, מערך המילים וPC שבו נעצרה פונקציית Write2memin. הפונקציה ממשיכה למלא את תמונת הזיכרון עבור כל המילים שלא נכתבו. במידה ויש מרווח, הפונקציה מרפדת באפסים את קובץ המוצא עד המילה האחרונה שנדרש לייצא

<u>רשימת פרמטרים:</u>

שם	ערך
MAX_LINE_LENGTH	500
MEMORY_SIZE	4096
MAX_LABEL_LENGTH	50
MAX_NUMBER_OF_LABELS	4096
DELIMITER	", #\t\n\r"
DELIMITER_C	", \t\n\r"

<u>פונקציונליות:</u>

התוכנה מבצעת שני מעברים על קובץ הקלט, במהלכן היא מפענחת את הטקסט, מעבירה אותו לייצוג המתאים לכתיבה בקובץ הזיכרון ולבסוף מבצעת כתיבה לקובץ הפלט. בשני המעברים התוכנה מתעלמת משורות ריקות, שורות המכילות הערות בלבד, באופן שיתן גמישות מלאה בכתיבת קובץ האסמבלי מבלי לפגוע בפונקציונליות.

המעבר הראשון: מטרתו לטפל ב-labels ו-word. (תוויות ומילים) באמצעות פונקציה בשם find_label_words_line המוזכרת לעיל. פונקציה זו סורקת את קובץ הקלט ומאתרת את הלייבלים ואת פקודות הword לפי המבנה הנדרש בפרויקט.

המעבר השני: במהלכו מתבצעת כתיבה לקובץ המוצא ע"י פונקציית write2memin המתוארת לעיל. הפונקציה לוקחת שורה מקובץ הקלט ומחלקת אותה לאסימונים. לפי סוג האסימונים הפונקציה מסווגת

את ההוראה לI או לR, ומבצעת את פעולת הכיתה בהתאם. הפונקציה מוודאת אם מדובר בתווית או במילה, ומספקת טיפול מתאים לכל מקרה. הפונקציה מסיימת בכתיבת כל המילים שנשארו למוצא בעזרת פונקציית עזר, ומייצרת את תמונת הזכרון הסופית.

חלק ג' - Simulator

תיאור התוכנה:

תפקידו של הסימולטור בפרויקט זה הוא לדמות פעילות שוטפת של מעבד SIMP הכוללת: **ביצוע הוראות** - בהתאם להגדרת ההוראות בפרויקט, **ממשק עם התקני קלט-פלט** - כתיבה וקריאה מהדיסק הקשיח, ממשק עם המסך המונוכרומטי, ממשק עם הצג 7segment וכן ממשק עם נורות LED, **טיפול בפסיקות השונות** - פסיקה פריודית (irq0), פסיקת הדיסק (irq1) ופסיקה חיצונית (irq2), **תיעוד הפעילות**- ביצוע מעקב (trace) אחרי הליך העבודה וכתיבה לקבצי מעקב אחר מצב הרגיסטרים הפנימיים, רגיסטרי החומרה ומחזורי השעון.

בדומה לאסמבלר, הסימולטור מומש בשפת C, בסביבת VS, בהנחה שקבצי הקלט תקינים.

קלט/פלט:

קובץ קלט	קובץ פלט
Memin.txt - תמונת זיכרון התחלתית, למעשה פלט של האסמבלר	Memout.txt- תמונת זיכרון **לאחר** הרצת הסימולטור
Diskin.txt - תמונת זיכרון התחלתית בדיסק הקשיח	Diskout.txt - תמונת הזיכרון בדיסק הקשיח **לאחר** הרצת הסימולטור
Irq2in.txt - רשימת מחזורי השעון בהם תתרחש פסיקה מספר 2	Cycles.txt - כמות מחזורי השעון שעברו **במהלך** הרצת הסימולטור
	Hwregtrace.txt - מעקב אחרי כתיבה או קריאה מרגיסטרי החומרה **במהלך** ריצת הסימולטור
	Trace.txt - מעקב אחרי תמונת הרגיסטריים הפנימיים **במהלך** ריצת הסימולטור
	Monitor.txt - קובץ מכיל שורות **הקסדצימליות** אשר מציגות תמונה שמוצגת על גבי המסך

	Monitor.yuv - קובץ מכיל שורות **בינאריות** אשר מציגות תמונה על גבי במסך **לאחר** ריצת סימולטור
	Leds.txt - קובץ מכיל שינויים (הדלקת נרות) במערך הleds **במהלך** ריצת הסימולטור
	Display7seg.txt - קובץ מכיל שינויים (שינוי בתצוגה הצג) בצג seven-segment **במהלך** ריצת הסימולטור
	Regout.txt - תמונת הרגיסטרים הפנימיים **לאחר** ריצת הסימולטור

תיאור קובץ Header:

<u>רשימת פרמטרים:</u>

שם	ערך
NUMBER_OF_INPUT_ARGS	16
MAX_LINE_LENGTH	500
NUMBER_OF_REGS	16
NUMBER_OF_IO_REGS	23
MONITOR_SIZE	256*256
MEMORY_SIZE	4096
MAX_NUM_OF_IRQ2	4096
DISK_NUM_OF_SECTORS	128
DISK_SECTOR_SIZE	128
DISK_SIZE	128*128
DISK_RW_TIME	1024

<u>מבני נתונים שהוגדרו:</u>

מבנה	פונקציונליות	שדות
simulator	מדמה חומרת סימולטור,	unsigned int PC;

	מאפשר אחיזה במערכי זיכרון ומעקב על סיגנלים השונים במהלך הריצה.	unsigned int regs[NUMBER_OF_REGS]; unsigned int; io_regs[NUMBER_OF_IO_REGS]; uint8_t monitor[MONITOR_SIZE]; int memory[MEMORY_SIZE]; int disk[DISK_SIZE]; unsigned int; cycles_of_irq2[MAX_NUM_OF_IRQ2]; unsigned int current_irq2; unsigned int disk_dcnt; bool handling_irq; bool run_en; int curr_monitor_index;
instruction	מדמה הוראה בודדת בשפת מכונה, מאפשר אחסון חלקים נפרדים של הוראה במבנה אחד לטובת גישה נוחה ומהירה יותר	unsigned int opcode :8 unsigned int rd :4 unsigned int rs :4 unsigned int rt :4 int imm :20 bool i_type

הערות על מבני נתונים:

1. זיכרון פנימי וזיכרון על דיסק קשיח מומשו בפרויקט ע"י מערכים של int בגודל מתאים לדרישות הפרויקט.
2. רגיסטרים (פנימיים וחומרה) ממומשים ע"י מערכי unsigned int בגודל המתאים לכל קבוצת הרגיסטרים.
3. משתנה עזר בוליאני run_en מאפשר לעצור לולאת while בתוך פונקציית run_simulator כאשר מתקבלת הוראת Halt.
4. משתנה עזר בוליאני handling_irq נותן אינדיקציה שהמעבד מטפל כרגע באחת הפסיקות.
5. משתנה עזר current_irq2 עוקב אחרי אינדקס עדכני במערך המחזורי שעון בהם תתרחש פסיקה irq2.
6. משתנה disk_dcnt מאפשר ספירה לאחור של 1024 מחזורי שעון בממשק עם הדיסק.

<u>פונקציות עזר:</u>

פונקציה	תיאור
load_data_from_input_file	טעינת מידע מהקובץ אל המבנה simulator. פונקציה זו

	תומכת בשלושת קבצי הקלט.
init_simulator	אתחול של הסימולטור.
run_simulator	הפונקציה המרכזית. כוללת לולאת while אשר מדמה איטרציות במעבד, כאשר כל איטרציה היא מחזור שעון בודד. מטרת הפונקציה - מעבר על ההוראות וביצוען תוך כדי טיפול בפסיקות וכתיבה לקבצים הנדרשים.
initialize_instruction	אתחול של ההוראה. הפונקציה נקראת בכל מחזור שעון כאשר מתקבלת הוראה חדשה.
*_write_output_file	סט של פונקציות לכתיבה לקבצי פלט, כל אחת מיועדת לקובץ נפרד.
disk_cmd_handler	פונקציית עזר, אשר מזינה את הרגיסטר command בדיסק בפקודה רלוונטית - קריאה או כתיבה. זאת במידה והדיסק פנוי.
disk_main_handler	פונקציה אשר בודקת האם הדיסק פנוי, ובמידה ולא מתבצע טיפול בפסיקה מספר 1. לאחר תום הזמן (1024 מחזורי שעון) מתבצע אתחול הרגיסטרים הרלוונטיים והדלקת אינדיקציה לפסיקה 1.
read_from_disk , write_to_disk	פונקציית מעטפת לכתיבה או קריאה לדיסק.
inc_clk	פונקציה אשר מטפלת בקידום מונה מחזורי השעון בסימולטור, נקראת בסוף כל איטרציה של הלולאה.
timer	פונקציית טיפול בפסיקה מספר 0
irq2	פונקציית טיפול בפסיקה מספר 2
irq_handler	פונקציה שמבצעת בדיקה שוטפת של סטטוס הפסיקות בסימולטור ובמידת הצורך מבצעת אתחול רגיסטרים רלוונטיים לטיפול בפסיקה.

<u>פונקציונליות:</u>

בהתאם ללולאת fetch-decode-execute עבודת הסימולטור מתחלקת לכמה שלבים:

1. **הכנת תשתית לעבודת הסימולטור**: פתיחת קבצי קלט/פלט לכתיבה או קריאה בהתאם לצורך, הקצאה דינמית של הזכרון למבנה simulator.

2. **אתחול הסימולטור**: אתחול השדות במבנה simulator. יש להדגיש שכלל הערכים המספריים והבוליאניים במבנה מאותחלים לאפס\false, וכן כלל המערכים נטענים בערכים המתאימים בהתאם לקבצי הקלט אותם מקבל הסימולטור (זיכרון התחלתי, דיסק התחלתי ורשימת מחזורי השעון בהם תתרחש irq2).
3. **הרצת סימולטור:** באמצעות פונקציה run_simulator. זו למעשה ליבת הסימולטור, פונקציה אשר עוברת על כלל ההוראות בזיכרון הראשי באמצעות לולאת while, ומבצעת אותן עד שנתקלת בהוראת Halt. בכל איטרציה של הלולאה, מתקיימות מספר פעולות:
   1. חילוץ הוראה מתוך זיכרון בכתובת PC, פענוחה וטעינתה לתוך מבנה instruction.
   2. בדיקת טריגרים לפסיקות - כתיבה לרגיסטרי status לפסיקות הרלוונטיות.
   3. כתיבה תמונת מצב הרגיסטרים הפנימיים לקובץ trace.txt טרם ביצוע הוראה.
   4. ביצוע הוראה בהתאם ל-opcode שחולץ מהזיכרון. יש לשים לב שחלק מההוראות משנות את הPC הבא בתור.
   5. כתיבה לקבצים hwregtrace, leds, display7seg במידה ומתקיימים תנאים לכתיבה (כלומר אינטרקציה עם אחד ההתקנים הללו).
   6. אתחול רגיסטרי חומרה ושינוי PC הבא במידת הצורך לטובת טיפול בפסיקות.
   7. קידום PC בהתאם לסוג הוראה, קידום מונה השעון.
4. **כתיבה לקבצי הפלט** הבאים לאחר סיום לולאת הריצה (כלומר לאחר קבלת halt):

   Memout.txt ,Regout.txt ,Monitor.txt ,Monitor.yuv ,Cycles.txt, Diskout.txt

   הכתיבה מתבצעת ע"י העתקה של ערכים ממערכי זיכרון מתאימים במבנה simulator לתוך הקובץ.
5. לאחר יציאה מפונקציית run_simultaor מתבצעת **סגירת כלל הקבצים** ושחרור הזיכרון שהוקצה בצורה דינמית.

חלק ד' - תכנות בדיקה בשפת Assembly

בפרויקט כתבנו ארבעה קבצי קלט באסמבלי שמטרתם לבדוק את פעולת הסימולטור.

1) Triangle.asm - בבדיקה זו התמקדנו בבחינת ממשק מול המסך, למעשה כתיבה לקבצים monitor.yuv ו-monitor.txt. במהלך הבדיקה בוצעה כתיבה המדמה משולש מלא המוצג במסך. מתוך קואורדינטות של שלושת הקודקודים חושבו אורכי הניצבים וערכים קבועים במשוואת הישר:

$$(y - y_A) = (y_C - y_A)/(x_C - x_A) * (x - x_A)$$

באמצעות שלושת הביטויים הללו בוצע מילוי של המשולש, זאת באמצעות האלגוריתם הבא - איטרציה על שורות הפיקסלים בניצב האנכי, כאשר לכל שורה מתבצעות איטרציות מילוי הפיקסלים בצורה אופקית תוך בדיקת תנאי חציית היתר לפי משוואת הישר:

$$(y - y_A)(x_C - x_A) <= (y_C - y_A) * (x - x_A)$$

במידה וזוהתה חריגה מתנאי החצייה, מתבצע דילוג לשורה הבאה של הפיקסלים.

2) Binom.asm - הבינום מומש באמצעות קריאות רקורסיביות המוגדרות לפי הנוסחה הרקורסיבית:

$$binom(n, k) = binom(n - 1, k - 1) + binom(n - 1, k)$$

כך שתנאי העצירה לכל מסלול רקורסיבי הוא: k=0 or n=k ואז נוסף 1 לסכום (כלומר התוכנה הגיעה למקרה יציאה). כאשר הארגומנטים התחלתיים נקראים מהזיכרון. בסוף הריצה הרקורסיבית התוצאה נשמרת בכתובת 0X102 כנדרש.

3) sort.asm - תוכנה שממיינת מערך של 16 מספרים באמצעות אלגוריתם מיון בעיות, באמצעות 2 לולאות מקוננות. הלולאה החיצונית רצה על כל המספרים פעם אחת, ונותנת בכל פעם את נקודת ההתחלה עבור הלולאה הפנימית. הלולאה הפנימית בודקת האם המספר הבא בסדר בזכרון קטן מהמספר הנוכחי, ואם כן מחליפה אותם. כך בעצם בכל איטרציה של הלולאה החיצונית אנחנו דואגים למיון המספר המתאים לאינדקס שלה, ובסיום הריצה נבטיח מערך ממוין בסדר עולה.

4) Disktest.asm - בדיקה זו בוחנת נכונות המימוש של ממשק מול הדיסק הקשיח. במהלך הבדיקה נעשית קריאה של שמונה סקטורים עוקבים מהדיסק (0-7), סכימתם וכתיבת התוצאה לסקטור התשיעי הבא אחריהם.

חלק ה' - סיכום

לסיכום, פרויקט ה-ISA כלל תכנון ויישום של אסמבלר וסימולטור בשפת C המקבלים קבצי קלט בשפת אסמבלי עבור מעבד ה-SIMP ומבצעים את ההוראות כנדרש. במהלך הפרויקט עסקנו בפרטים המורכבים של ארכיטקטורת המעבד, ארגון הזיכרון, פעולות קלט/פלט וטיפול בפסיקות. על ידי פיתוח סימולטור המדמה את אופן הפעולה של רכיבים אלו, רכשנו תובנות רבות על אופן הפעולה הפנימי של מעבד מבוסס RISC. האסמבלר אפשר לנו לתרגם קוד אסמבלי באופן מדויק להוראות קריאות בשפת מכונה, מה שאפשר אינטראקציה חלקה עם מעבד ה-SIMP המסומלץ. לאחר מכן, הסימולטור ביצע את ההוראות הללו ואיפשר לנו לסמלץ את מחזור ה-fetch-decode-execute ולצפות בהתנהגות המעבד בתרחישים שונים כגון מיון מספרים, בעיות מתמטיות וממשק עם התקני קלט פלט. לסיכום, הפרויקט היה כלי משמעותי בהעמקת הבנתנו של מושגי הקורס.